Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Mapování prostředí pomocí kvadkoptéry

Environment Mapping Using a Quadcopter

2014 Miroslav Meca

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Miroslav Meca**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Mapování prostředí pomocí kvadkoptéry
Environment Mapping Using a Quadcopter

Zásady pro vypracování:

Cílem práce je implementace základních kroků pro 3D rekonstrukci scény z kamery, která je umístěna na kvadrikoptéře. Motivací pro práci je určování polohy kvadrikoptéry v prostoru, která je však příliš složitou pro bákalářskou práci.

Ve své práci proveďte:
1. Nastudujte algoritmy pro detekci význačných bodů v obraze.
2. Aplikujte algoritmy pro detekci význačných bodů na data z kamery kvadrikoptéry.
3. Proveďte spárování význačných bodů z předchozího bodu.
4. Výsledky naměřte a řádně zhodnoťte.

Seznam doporučené odborné literatury:

[1] Lowe, D.: Object recognition from local scale-invariant features. Proceedings of the International Conference on Computer Vision 2. pp. 1150-1157 (1999)
[2] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: SURF: Speeded Up Robust Features. Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346-359 (2008)

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Gaura**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 5. května 2014 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. května 2014 .....

Rád bych poděkoval mému vedoucímu Ing. Janu Gaurovi za věnovaný čas, cenné rady a trpělivost, kterou se mnou měl. Spolužákovi Radku Simkaničovi, který mi půjčil počítač, abych na něm mohl pracovat, když mi odešel notebook. A na závěr své matce za podporu během studia.

Abstrakt

Tahle bakalářská práce je zaměřená na mapování prostředí pomocí kvadkoptréry. Jako zástupce kvadkoptéry byla vybrána Parrot ArDrone2. Práce popisuje způsob ovládání a získávání dat kvadkoptéry a algoritmy pro detekci význačných bodů v obraze a jejich spárování.

Klíčová slova: Kvadkoptéra, Parrot Ar.Drone2, SIFT, RANSAC, SURF, navdata, SDK, OpenCV, C, Python

Abstract

This bachelor thesis is focused on mapping environment using quadcopter. As a representative of quadcopter was chosen Parrot ArDrone2. The work describes a method of control and data acquisition quadcopter and algorithms for the detection of key points in the image and their pairing.

Keywords: Quadcopter, Parrot Ar.Drone2, SIFT, RANSAC, SURF, navdata, SDK, OpenCV, C, Python

Seznam použitých zkratek a symbolů

SIFT	–	Scale-Invariant Feature Transform
SURF	–	Speeded Up Robust Features
BBF	–	Best-bin-first algoritmus
RANSAC	–	Random Sample Consensus
LiPo	–	Lithio-polymerová baterie
MEMS	–	Micro-Electro-Mechanical Systems
fps	–	Frame Per Second - počet snímků za sekundu
SDK	–	Software development kit
Ω	–	Úhlová rychlost
Δ	–	Změna otáček vrtule
TCP	–	Transmision Control Protocol
UDP	–	User Datagram Protocol
UAV	–	Unmanned Aerial Vehicle - Bezpilotní letoun
CR	–	Carriage return - nový řádek
WiFi	–	Wireless Fidelity
LED	–	Light-Emitting Diode
ROS	–	Robot Operation System
USB	–	Universal Serial Bus
DoG	–	Difference of Gaussians
LoG	–	Laplacian of Gaussian
Navdata	–	Navigační data

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

Úvod

Kvadkoptéry, tedy vrtulníky se čtyřmi motory, patří k bezpilotním letounům. V poslední době se značně rozšířili díky masové výrobě a možnosti létat venku i vevnitř. Zmenšené vrtule díky čtyřem rotorům, vytváří při letu nižší kinetickou energie a to snižuje riziko zničení letounu. Aplikace pro kvadkoptéry jsou různorodé, od hraní her až po sledování objektů. Více o kvadkoptérách v 1 kapitole.

Má bakalářská práce je zaměřená na model kvadkoptéry Ar.Drone2 od firmy Parrot, kterou je možno připojit k počítači pomocí WiFi spojení. Ar.Drone2 je vybaven sadou senzorů, dvěma kvalitními kamerami a palubním počítačem běžícím pod GNU/Linuxem. Princip létaní, komunikace a vybavení kvadkoptéry bude popsáno v 2 kapitole.

Kameru kvadkoptéry je také možno využít pro mapování prostoru. Když kvadkoptéra prolétává prostorem, řídicí software vytváří 3D model prolétnuté scény. Toto je velmi náročná operace jak po teoretické tak po praktické stránce, a proto jsem se ve své bakalářské práci zaměřil na implementaci základních operací pro 3D rekonstrukci scény z kvadkoptéry.

Hlavními cíli práce tedy bylo zprovoznění kvadkoptéry s počítačem, získaní dat z videokamery, ovládání kvadkoptéry joystickem, detekce význačných bodů pro 3D rekonstrukci a získaní korespondence mezi dvěma po sobě jdoucích obrazů z kvadkoptéry. Zpracování obrazu z kamery bude popsáno v 4 kapitole.

1 Kvadkoptéra

Lze se setkat s označením *kvadrokoptéra*, *kvadrikoptéra*, *quadrokoptéra*, *quadkoptéra*, *quadrotor* a další. Spadají do kategorie bezpilotní zařízení. Kvadkoptéry používají elektronický kontrolní systém a senzory ke své stabilitě.[3]

1.1 Bezpilotní letouny

Bezpilotní letoun, v angličtině Unmanned Aerial Vehicle (UAV). Letou může být ovládán ze země anebo automaticky, třeba předem určenou cestou. Letoun může obsahovat i různé pozemní stanice a další přídavné zařízení.[1, 2]

Bezpilotní letouny se uplatňují ve více oborů. Například ve vojenském, výzkumném a civilním. Jejich využití se dá najít v takových situacích, které jsou nebezpečné pro lidský život. Navíc se zvyšuje u letounu maximální přetížení na více jak 30G, které konstrukce dokáže vydržet, zatímco člověk vydrží kolem 10G. Jejich celková velikost je menší než u letounů s pilotem a tím pádem hůře zpozorovatelné.[1, 2]

1.2 Popis kvadkoptéry

Kvadkoptéra je zajímavý stroj, jeho způsob létání by se dal přirovnat k vrtulníku. Je to bezpilotní letoun, který je poháněn čtyřmi rotory, ke kterým jsou pevně přidělané šikmé vrtule umístěné v jedné rovině do dvou sad. Jedna sada (dvě vrtule) je přidělaná ve směru hodinových ručiček a druhá sada proti směru. Těmito vrtulemi se kontroluje vztlak a točivý moment. Konečný pohyb je dosažen změnou rychlosti jednoho nebo více rotoru a tím se mění momentální zatížení, tah a zdvih.

1.3 Výhody kvadkoptér oproti vrtulníku

Díky velikosti a obratnosti můžou být použity venku i vevnitř. Kvadkoptér oproti srovnatelně zmenšenému vrtulníku má menší průměr vrtulí a to díky čtyřem rotorům. Tím získáváme menší kinetickou energii při letu. To snižuje poškození rotorů při střetnutí a navíc umožňuje u malých letounu uzavřít důležité součástky a to i rotory samotné do ochranného obalu. Umožňuje to bezpečně létat i v náročném prostředí. Dále kvadkoptéra nevyžaduje mechanické vazby pro změnu úhlu rotoru ke stoupání při otáčení, to zjednodušuje konstrukci a údržbu.

2 Ar.Drone2

Tato kapitola se zabývá Ar.Drone2, jejími parametry, zařízeními a komunikací.

Ar.Drone2 je rádiem, či WiFi řízená létající kvadkoptéra postavena francouzskou firmou Parrot. Balení obsahuje kvadkoptéru Ar.Drone2, kryty pro vnitřní a venkovní létání, reflexní samolepky, LiPo baterii, adaptér k baterii a manuál.[3]

Tato kvadkoptéra běží pod GNU/Linuxem 2.6.32 a má v sobě zabudovanou automatickou stabilizaci letu. Pokud vypadne WiFi připojení s klientem do pár vteřin automaticky přistane.[4, 6]

2.1 Konstrukce

Konstrukce Ar.Drone2 vydrží více než u předešlé generace. Došlo k vyztužení nejvíce zatěžovaných části. Je použita pěna proti otřesům k snížení vibrací u vnitřních dílů.[3, 4, 6]

Ochranné kryty pro vnitřní a venkovní létaní jsou vyrobeny z tvrzeného polystyrénu. Nosný kříž je zpevněn karbonem. Vrtule a některé další díly jsou z obyčejného plastu. Obrázek 2.5 schématu Ar.Drona2 s kryty.[3, 4, 6]

2.2 Vybavení Ar.Dronu2

2.2.1 Vrtule a motory

Jak již bylo zmíněno, Ar.Drone2 má 4 vrtule. Dvě šikmé na pravou stranu a dvě na levou. Vrtule mají průměr 20 cm. Motory mají hřídel z tvrdé oceli a samomazné bronzové ložiska. Díky tomuto vybavení dokáže letět až 18 km/h.[3, 6]

Každou vrtulí otáčí samostatný elektromotor, z důvodu požadavku na řízení otáček každé vrtule zvlášť. Používají se čtyři 15 W rotory poháněných střídavým napětím 11,1 V. Motory můžou mít až 28500 otáček za minutu. Vrtule nejsou připevněny přímo na hřídel motoru, ale jsou poháněny přes ozubená kola.[3]

Rychlost motorů je řízena 8-bitovým mikro kontrolérem 8MIPS AVR CPU a 10-bitovým analogově digitálním převodníkem a dvou barevnou LED signalizací. Ar.Drone2 automaticky rozpozná typ připojených motorů a automaticky upraví ovládací prvky motoru. Detekuje také, zda všechny motory fungují, či nenarazili na překážku. V případě, že rotující vrtule narazí na jakoukoliv překážku a je blokována, Ar.Drone2 zastaví v takovém případě všechny motory.[3, 4, 6]

2.2.2 LiPo Baterie a nabíječka

Jediný zdroj energie Ar.Dronu2 je tří článková lithium-polymerová baterii, která se značí LiPo. Tato technologie vychází z lithio-iontových akumulátorů. V baterii vzniká elektrická energie chemickou reakcí.[3, 7]

Napětí LiPo článků se pohybuje mezi 2.7 V až 4.2 V. Tyhle hodnoty nesmí být překročeny, jinak se baterie rychle zničí.[7]

Baterie používaná Ar.Drone2 má celkovou kapacitu 1000 mAh a napětí 11,1 V. Při nabitém stavu je to 12,5 V a vybitém kolem 9 V. Průměrná doba létání s nabitým akumulátorem je 14-16 minut. Ar.Drone2 monitoruje stav baterie a zobrazuje ho v procentech (100% když je nabitý a 0% když je vybitý). Když Ar.Drone2 detekuje slabý stav baterie, první pošle varování uživateli a poté automaticky přistane na zem.[3, 4, 6]

Baterie se nabíjí přes speciální nabíječku dodanou s Ar.Dronem2. Nabíječka se stará o nabíjení každého článku zvlášť. Signalizace nabití je přes LEDky v nabíječce. Kde červená barva značí, že je v síti a není plně nabitá, zelená barva značí plné nabití. Celkový doba nabíjení trvá až 90 minut.[3, 4, 6]

2.2.3 Kamery

Letoun je vybaven dvěma kamerami. Horizontální, která směřuje dopředu, a vertikální, která je umístěna zespodu a snímá podlahu. Video je kódováno do formátu H264 a ukládá se jako JPEG.[3]

Obě kamery snímají na čip CMOS se zorným úhlem 92°. Ar.Drone2 automaticky vytváří kódování a stream obrazu pro klientské zařízení. Horizontální kamera má rozlišení 360p (640x360) nebo 720p (1280x720), a frekvenci snímání může mít 15 fps nebo 30 fps. Vertikální kamera má rozlišení QCIF (176x144) anebo QVGA (320x240), s frekvencí snímání 60 fps.[3, 4, 6]

Lze získávat obraz pouze z jedné kamery a ne z obou zároveň. Pokud chceme-li získat obraz z druhé kamery, můžeme během provozu poslat příkaz, který nastaví posílaní dat z druhé kamery a naopak.

2.2.4 Senzory, měřiče a elektronika

Letoun má několik pohybových senzorů, umístěných v centrálním trupu. Pod spodním krytem najdeme dvě desky plošných spojů. Jednu bychom mohli označit jako navigační, protože obsahuje gyroskop pro 3 osy, magnetometr, akcelerometr, ultrazvukový výškoměr a vzduch tlakový snímač. Údaje ze senzorů zpracovává mikroprocesor.[4, 6]

Tyto senzory usnadňují stabilizaci letu a zlepšují funkčnost. Dále umožňují vytvořit autonomitu, aby mohl Ar.Drone2 vyhodnocovat získané údaje a sám létat bez pomocí pilota. K zjištění výšky slouží výškoměr, je důležité znát i polohu kvadkoptéry, k tomu slouží gyroskopy a akcelerometry. Ar.Drone2 využívá MEMS technologie.

2.2.4.1 MEMS Tato zkratka znamená Micro-Electro-Mechanical Systems, tahle technologie spojuje polovodičové a mechanické části do jednoho integrovaného obvodu, na křemíkové bázi. Tyto obvody v dnešní době mají velikost v řádu jednotek milimetrů. Produkty MEMS jsou především pohybové senzory (gyroskopy, akcelerometry,...).[8]

MEMS-based jsou úplně stejné zařízení jako MEMS, pouze mají navíc malé topné těleso ve spodní části, které ohřívá vzduch uvnitř součástky a tím vychyluje plovoucí části mimo střed.[8]

2.2.4.2 Výškoměr Výškoměr slouží na bezkontaktní měření vzdáleností. V praxi se používá optického paprsku anebo zvukového vlnění. Obě metody pracují na stejném principu. Vyšle se signál a ten se odráží od překážky a dopadá zpět na snímač. Na základě doby mezi vysláním a přijmutím signálu se určí vzdálenost. Ar.Drone2 používá ultrazvukového výškoměru s frekvencí, kterou lidské ucho neslyší. Umístění ultrazvuku je na spodní straně.[3]

2.2.4.3 Gyroskop Gyroskop je zařízení pro měření nebo udržení orientace, poskytuje údaje o úhlové rychlosti (otáčení) a naklonění. V kartézském souřadném systému je možné rotovat objektem ve třech osách. MEMS gyroskopy zastává myšlenku Foucaltova kyvadla. Foucaltovo kyvadlo souvisí s rotací země a její zdánlivou sílou, které se říká Coriolisova síla.[9]

Věta 2.1 *Síla působící na hmotný bod pohybující se rychlostí v v soustavě rotující úhlovou rychlostí* Ω *je Coriolisova síla.*

Z věty 2.1 vyplývá, že Coriolisova síla je úměrná rychlosti a směru otáčení, a tím můžeme určit úhlovou rychlost.

MEMS gyroskop se skládá ze vzduchových kondenzátorů, automatické vynulování, teplotní čidlo, měřící plošky, pružiny, na kterých je uvnitř rámu upevněn pohybující se objekt o přesně dané hmotnosti. V přesných intervalech je elektronicky pohybováno tímto objektem. Coriolisova síla způsobí stlačení pružin rámu a posun měřících plošek. Ty změní kapacitu vzduchových kondenzátorů. Dle zjištěných hodnot jsme schopní určit úhlovou rychlost.[8, 9]

2.2.4.4 Akcelerometry MEMS akcelerometr je přístroj, který měří akceleraci a statické zrychlení daného předmětu, nejčastěji pomocí změn kapacity vnitřního kondenzátoru, vyvolané vlivem síly vzniklé zrychlením. Akcelerometrem se také zjišťují vibrace a ke stabilizaci letu.[8, 10]

MEMS akcelerometr se skládá z kondenzátoru, demodulátoru a zesilovače. Kondenzátor akcelerometru se pak skládá z děliče obdélníkového signálu, dvou pevných desek a pohyblivého nosníku, který je spojen mikrochemickýma polo-křemíkovýma plovoucíma pružinami na povrchu křemíkového monokrystalu. To vše je v uzavřeném pouzdře.[8, 10]

Pod vlivem vnějšího zrychlení dojde posunutí pohyblivého nosníku, pružiny a to umožní pohyb monokrystalu. Tím poskytují mechanický odpor, který se projeví také na změně dělícího poměru kondenzátoru. Na výstupu děliče se objeví obdélníkový signál. Signál nese informaci o síle a směru zrychlení. Vyhodnocení signálu provádí demodulátor a jeho výstup je přiveden na vstup zesilovače. Vnitřní zpětná vazba vrací pohyblivý nosník zpátky do neutrální pozice a to pomoci elektrostatických sil. [8, 10]

2.2.5 WiFi síť a připojení

Druhá deska pod spodním krytem se označuje jako základová. Obsahuje WiFi modul, procesor, operační paměť, kamery, napájení, ovládání motorů a další. Ar.Drone2 podporuje WiFi standarty 802.11b/g/n.[3]

Ar.Drone2 při zapnutí automaticky vytvoří síť s názvem *ardrone2_xxxx* a spustí DHCP server a nastaví si IP adresu (typicky 192.168.1.1). Připojování na něj probíhá stejně jako, kdybyste se chtěli připojit na jakoukoliv bezdrátovou síť. Po připojení je DHCP serverem na Ar.Dronu2 klientovi přiřazena IP adresa 192.168.1.2-255. Poté může připojené zařízení posílat požadavky na porty 5554, 5555, 5556 a 5559, a tím získat požadované data.[3]

2.2.6 Zbylý hardware

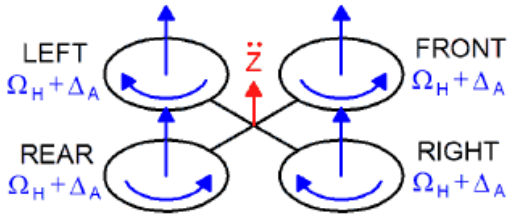
Ar.Drone2 obsahuje 32-bitový procesor ARM Cortex A8 s frekvencí 1 GHz, operační paměť 128 MB (1 Gbit) DDR2 RAM 200 MHz a USB 2.0 port pro připojení USB flash paměti anebo GPS příslušenství.[3]

2.3 Princip létání

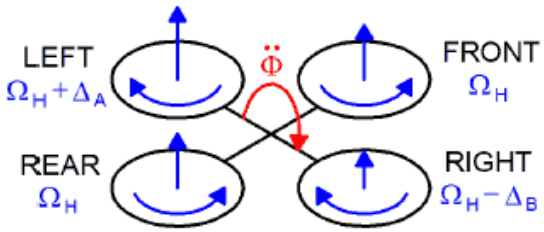
Princip létání kvadkoptéry je velmi specifický a složitý, i když připomíná ovládání vrtulníku. Má ovšem všechny své vrtule v jedné rovině. Veškerý pohyb se děje na základě změny otáček jedné nebo více vrtulí. Jak již bylo zmíněno, dvě z vrtulí se točí po směru hodinových ručiček a dvě proti. To proto, aby byl vyrovnán točivý moment.[3]

Změny rychlosti vrtulí se projevují pohybem kvadkoptéry. V následujícím textu a obrázcích (2.1, 2.2, 2.3 a 2.4) lze vidět základní princip pohybu letounu. V těchto obrázcích je Ω úhlovou rychlostí a Δ změna otáček vrtule.

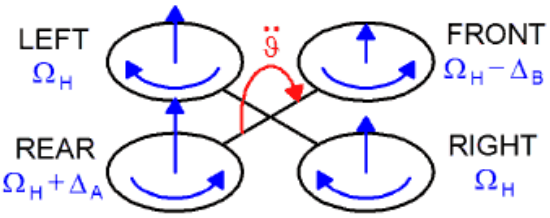
- Obrázek 2.1 je pro stoupání. Zvyšuje se počet otáček na všech vrtulí stejně. Pro klesání by se snižoval.
- Obrázek 2.2 je pro let vpravo. Na levé vrtuli se zvýší otáčky a na pravé vrtuli se sníží otáčky, to proto, aby byla zachována rovnováha točivých momentů. Výsledný efekt je takový, že se stroj nakloní doprava a letí vpravo, aniž by spadnul. To samé platí, kdybychom chtěli letět na opačnou stranu, stačí pouze zesílit otáčky na pravé vrtuli a na levé zeslabit.
- Obrázek 2.3 je pro let dopředu. Stejný princip jako u předchozího obrázku, pouze se nezesiluje a nezeslabuje levá a pravá vrtule, nýbrž zadní a přední.
- Obrázek 2.4 je pro otáčení proti směru hodinových ručiček. Využívá se točivých momentů. Přední a zadní vrtule se točí stejným směrem a to proti směru hodinových ručiček, zatímco levá a pravá se točí po směru hodinových ručiček. Aby se udržela výška, musí se zvýšit rychlost u jedné ze skupin vrtulí a u druhé se sníží. Podle toho jaká skupina vrtulí se sníží a jaká zvýší se, buď otáčí doprava anebo doleva.



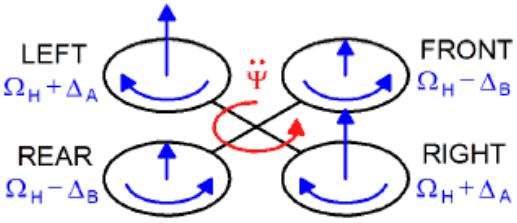
Obrázek 2.1: Stoupání a klesání [3]



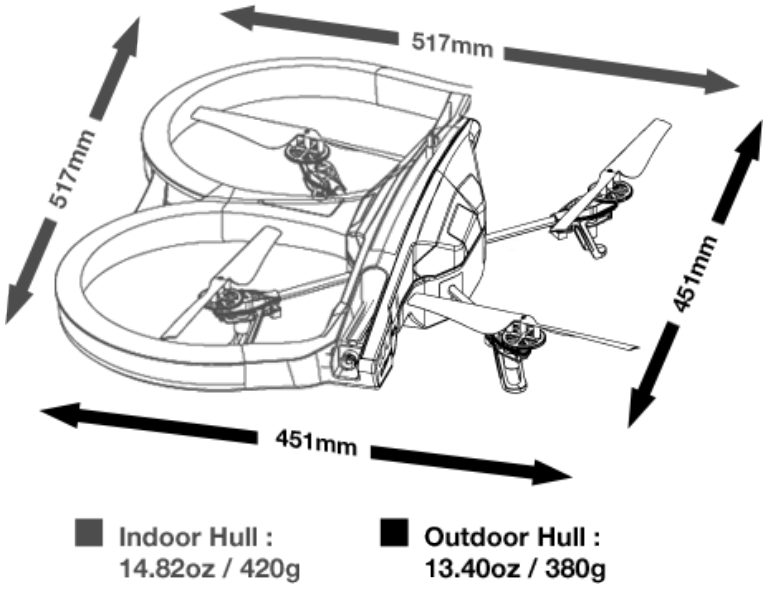
Obrázek 2.2: Let vlevo, vpravo [3]



Obrázek 2.3: Let vpřed, vzad [3]



Obrázek 2.4: Zatáčení doleva, doprava [3]



Obrázek 2.5: Schéma ArDrona2 [5]

2.4 Komunikace

Ar.Drone2 je přístupovým bodem a lze se na něj přes WiFi připojit. Připojení k WiFi síti drona je probíráno v 2.2.5 kapitole.

Ovládání Ar.Drona2 se provádí pomocí následujících komunikačních služeb:

- **AT Commands** slouží k ovládání drona na UDP portu 5556.2.4.1
- **Navdata** jsou to navigační data, která se odesílají klientovi na UDP port 5554. Navdata obsahuje údaje o pozici, rychlosti, stav baterie, otáčky motoru, pozici kamery a podobně.2.4.2
- **Video stream** je vysílán na TCP/UDP portu 5555.
- **Control port** je na TCP portu 5559 a to pro přenos kritických dat. Tato služba se používá například pro přenos potvrzování konfiguračních příkazů. Když nepřijde potvrzení, pošlou se znovu.

2.4.1 AT Commands

AT příkazy se posílají UDP protokolem na portu 5556, tyto příkazy slouží ke konfiguraci a ovládání Ar.Drona. Je zapotřebí posílat Ar.Dronu neustále příkazy, jestli nedostane žádný příkaz déle jak 2 sekundy, je to bráno jako ztráta WiFi připojení a provede se automatické přistání. Pro plynulý pohyb je nutno příkazy posílat minimálně každých 30 ms.[3]

AT příkazy jsou textové řetězce, tedy sekvence ASCII znaků, proto je nutné i čísla převést na řetězce. Tyto příkazy musí začínat řetězcem *AT**, pak následovat názvem příkazu s rovnítkem, za kterým je pořadové číslo a případně seznam argumentů oddělených

čárkami. Příkazy musí končit znakem nového řádku *"\r"*, označovaný jako <*CR*>. Jeden UDP paket může obsahovat více příkazu oddělených novými řádky. AT příkazu je zobrazený v 2.1 příkladu. Maximální délka celkového příkazu nesmí přesáhnout 1024 znaků, jinak se cela sada příkazu v UDP paketu zamítne. Nesprávné příkazy jsou ignorovány.[3]

Příklad 2.1
AT*PCMD_MAG=21625,1,0,0,0,0,0,0<CR>AT*REF=21626,290717696<CR> ■

Pro zabránění zpracování starých příkazů, je pořadové číslo spojené s každým AT příkazem. ArDrone neprovádí příkazy s menším pořadovým číslem, než měl předchozí AT příkaz, výjimkou je příkaz *COMWDG*, který nuluje číslo sekvence. Pořadové číslo je nastaveno na 1 pokaždé, když se klient odpojí od portu AT Commands UDP.[3]

Seznam AT příkazů

- **REF** - má 1 argument, podle kterého se provede buď přistání, či vzlétnutí, vypnutí a zapnutí přívodu energie k rotorům a pohotovostní vypnutí příkazů
- **PCMD** - má 5 argumentů, dle kterých se pak ArDrone pohybuje
- **PCMD_MAG** - má 7 argumentů, s tímto typem příkazu získáváme absolutní kontrolu nad pohybem ArDrona
- **FTRIM** - bez argumentů, vynulování hodnot gyroskopu, ArDrone musí ležet na zemi (rovině)
- **CONFIG** - má 2 argumenty, nastavování ArDrona pomocí klíčů a hodnot
- **CONFIG_IDS** - má 3 argumenty, identifikátory pro CONFIG příkazy
- **COMWDG** - bez argumentů, restart "watchdogu"pro komunikaci a změny sekvence pořadových čísel
- **CALIB** - 1 argument, žádost o kalibraci magnetometru, ArDrone musí být ve vzduchu

2.4.2 Navigační data (Navdata)

Navdata jsou navigační data, které odesílá řídící aplikace na UDP port 5554. Navdata předávají informace o stavu ArDrona a to co 5 ms. V navigačních datech jsou obsaženy informace o hodnotách senzorů a stav kvadkoptéry.

Po spuštění Ar.Drona, ani po připojení klienta se navdata neposílají. Je zapotřebí poslat paket na UDP port 5554, a jeho datová část musí obsahovat následující 4 byty: $00000001_{(16)}$. Většinou se to provádí při inicializaci. Pak Ar.Drone posílá na tento port pakety, nicméně stále neobsahuje užitečná data. K získání užitečnějších dat musíme poslat AT příkaz *CONFIG* na port 5556, třeba s parametry *"general:navdata_demo","TRUE"*. Celá podoba příkazu je ve 1 výpisu. Nejdříve restartujeme sekvenci a pak nastavíme jaké navdata se mají posílat.[3]

```
AT*COMWDG=1\r
AT*CONFIG=2,"general:navdata\_demo","TRUE"\r
```

Výpis 1: Zapnutí posílaní užitečných Navdat.[3]

Navdata vždy obsahují hlavičku, stav kvadkoptéry, pořadové číslo, stav činnosti, id paketu a jeho velikost. Hlavička má velikost 4 bytů, je to 32bitové číslo, které má vždy stejnou hodnotu. Stav kvadkoptéry je opět 32bitové číslo a každý bit má svůj význam, určuje stav patřičného zařízení. Sekvence je 4 bytové celé kladné číslo, stav činnosti je také 4 bytové kladné číslo, které nás informuje o tom, co Ar.Drone zrovna provádí. ID a velikost je kladné číslo každé o 2 bytech. Navigační data jsou posílaná pomocí způsobů *Little endian*.[3]

Existují dva způsoby zapisování dat do paměti. A to *little endian* a *big endian*. Little endian se používá častěji a to díky velkému rozšíření procesoru Intel, který tento způsob využívá. Little endian zapisuje byte s nejnižším významem na nejnižší adresu v paměti a za něj se ukládají další byty. Big endian používá opačný způsob ukládání dat do paměti, a to, že se na nejnižší adresu uloží nejvíce významný byte a za ním ostatní.

3 Vývojové prostředí pro Ar.Drone

Pro vývoj Ar.Drona se může použít více vývojových nástrojů. Dva nejpoužívanější jsou ROS a ARDRone_SDK. Já si vybral ARDrone_SDK 2.0.1.

Zkratka SDK znamená software development kit, někdy se označuje také jako "devKit". Je to sada softwarových vývojových nástrojů, které umožňují vytvářet aplikace pro určitou softwarovou a hardwarovou platformu. SDK zahrnuje ukázkové kódy a podpůrné technické dokumentace. Balíčky jsou většinou k dispozici zadarmo.[11]

Já používám SDK balíček *ARDrone_SDK_2_0_1*, který uvolnil Parrot. SDK je k dispozici pro iOS, Android, Linux i Windows, používám se programovací jazyk C.

3.1 SDK na Ubuntu

Instalace SDK na Ubuntu není nijak složitá. Stačí udělat tyhle následující kroky.

1. Nejprve si stáhneme samotný balíček, třeba z tohoto odkazu `https://projects.ardrone.org/attachments/download/514/ARDrone_SDK_2_0_1.tar.gz`. A poté ho rozbalíme do složky, kde s ním budeme chtít pracovat. Adresář má tuto strukturu.[11]
   - ARDroneAPI.dox - soubor doxygen, pro generování dokumentace
   - ARDroneLib - ArDrone knihovny(komunikace s dronem, video kodeky, atd.)
   - ControlEngine - soubory specifické pro iPhone
   - Docs - složka, kde je vytvořena dokumentace
   - Examples - složka obsahující ukázky pro různé platformy
   - GoogleAPI
2. Ve složce */ARDroneLib/Soft/Build* změníme v souboru *custom.makefile* u *USE_LINUX* z *"no"* na *"yes"*. A provedeme *"make"*.
3. V případě problému spustíme skript ve stejné složce s názvem *check_dependencies.sh*, tento skript zkontroluje zda máme všechny potřebné balíčky a knihovny stáhnuté a nainstalované, pokud nějaký chybí nainstaluje je.
4. To je vše, co se týká instalace SDK. Pokud vše proběhlo v pořádku, můžeme SDK již využívat a pracovat s ním.

3.2 Sestavení příkladů

Sestavení příkladových projektů je dobré k tomu, abychom si otestovali, zda máme opravdu všechny potřebné balíčky nainstalované a zda může počítač komunikovat s dronem.[11]

1. Nejprve zkontrolujeme, zda máme všechny balíčky, to provedeme spuštěním skriptu *ARDroneLib/Soft/Build/check_dependencies.sh*, musí být zapnutý s oprávněním roota. Zobrazí se OK, pokud jsou všechny balíčky nainstalované.

2. Zkontolujeme v souboru */ARDroneLib/Soft/Build/custom.makefile* zda máme u *USE_LINUX="yes"*.

3. Provedeme *"make"* ve složce s příklady */Examples/Linux/*

4. Pokud se nepovedlo sestavit příklady, může to být ještě tím, že chybí některé z flagů v makefilu. Otevřeme si tedy soubor */Examples/Linux/Makefile* a přidej flagy, které chybí: *GENERIC_LIBS+=* přidáme následující flagy: *-liw -lpc_ardrone -lgthread-2.0 -lgtk-x11-2.0 -lrt -lcairo -lxml2 -ludev -lswscale -lSDL -lm*

5. Zapneme Ar.Drone2 a připojíme se na jeho síť. Pak můžeme spustit některých z příkladu, který je po přeložení umístěn ve složce */Examples/Linux/Build/Release/*

3.3 Obsah knihovny

ARDrone_SDK knihovna je k dispozici jako open-source. Důležitou součástí jsou určeny funkce pro ovládání ArDrona, které mohou být využity při vlastní aplikaci, s celým frameworkem ARDroneLIB a ARDroneTool, případně implementovat svůj vlastní Framework podle specifikací AT Commands, navdat a video streamu v 2.4 kapitole.

SDK umožňuje snadno psát aplikace na vzdálené ovládání ArDrona z jakéhokoliv osobního počítače s WiFi, zařízení Apple iOS, či ze zařízení Androidu. Nicméně SDK nepodporuje přepisování zabudovaného softwaru v ArDronu, což znamená, žádný přímý přístup k hardwaru.[3]

Složka *ARDroneLib* v SDK obsahuje:

- **SOFT** - kde jsou specifické kódy, včetně hlavičkových souborů popisující komunikační struktury, sada nástrojů pro snadnou správu ArDrona, jako jsou AT příkazy a vlákna pro příjem Navdat, videa a inicializaci hlavního vlákna
- **VLIB** - knihovna ArDrone 1.0 pro příjem a dekódování video streamu
- **FFMPEG** - kompletní balík FFMPEG knihovny, se skripty pro aplikace Ar.Drone 2.0
- **ITTIAM** - překompilované, vysoce optimalizované video dekódovací knihovna pro iOS a Android aplikace
- **VPSDK** - sada univerzálních multiplatformních knihoven. Obsahuje funkce pro alokaci paměti, řízení vláken, komunikační funkce pro WiFi a Bluetooth, správu potrubí videa a další

3.4 Joystick PS3

K ovládání drona budeme potřebovat nějaké vstupní zařízení. SDK sice nabízí strukturu a funkce pro vstupní zařízení ke komunikaci s dronem, nicméně ARDrone_SDK_2_0_1 špatně zachytává zmáčknuté klávesnice. Šlo by to opravit použitím OpenCV pro zachycení zmáčknutí klávesnic anebo využít část programu v příkladech pro ardrone_sdk, kde je vytvořen konečný automat pro joystiky typu PS3, Logitech a Radio GP. Měl jsem k dispozici joystick playstation 3 a proto jsem se ho rozhodl využít.

3.4.1 Instalace joystiku

K instalaci budeme muset nainstalovat potřebné ovladače a přidat do repository záznam. Joystik musí být zapojený do USB, příkazem v konzoli *lsusb* se vypíšou všechny zařízení připojené do USB sběrnice. Najdeme joystik od playstationu a jeho device_id použijeme. Jak provedeme všechny příkazy z 2 výpisu, restartujeme počítač. Joystik je nainstalován. [12]

```
sudo apt-get install libusb-dev libusb-0.1-4 xserver-xorg-input-joystick
sudo add-apt-repository ppa:falk-t-j/qtsixa
sudo apt-get update
sudo apt-get install qtsixa
lsusb
sudo sixpair XXXX:XXXX #(device ID)
sudo shutdown -r now
```

Výpis 2: Příkazy k instalaci PS3 joystiku

4 Zpracování obrazu

Vývoj techniky a softwaru, který získává informace ze zachycených obrazů, se nazývá počítačové vidění. Nejčastěji se používá k detekci některých předmětů, či lidských obličejů. Za poslední roky se tento obor dostal na lepší úroveň, velké zásluhy na tom má snadnější dostupnost kvalitního hardwaru, jako jsou kamery a laser skenery a výpočetní technika s dostačujícím výkonem, ale také jednodušší dostupnost balíčků knihoven, jako je například OpenCV.

4.1 Detekce význačných bodů

Důležitou součásti zpracování obrazu je detekce význačných bodů, které nám dávají základní informace a vybírají vhodné body pro další zpracování. Důležitou vlastností detektorů je stabilita a opětovná lokalizovatelnost. Je nutné body nalézt opakovaně a to i po změnách a deformacích.

Algoritmy pro detekci význačných bodů se neustále vyvíjejí. Vznikla hromada detektorů s různými vlastnostmi. Například Harrisův detektor, byl vytvořen v roce 1988 a vychází z Moravcova detektor. Moravcův detektor byl jednoduchý, výpočetně nenáročný, ovšem nestabilní a náchylný na šum. Zatímco Harrisův již byl odolný na šum, rotaci a posun, ale nedokázal si stále pomoct s měřítkem.[13]

K nejdůležitější výzkumy zabývajících se reprezentací obrázků nezávislých na měřítku, patří výzkumy pánů Linderberg a Lowe. Pomocí převodů dat z obrázku do frekvenčních oblastí, jsou struktury klíčových bodů tvořeny různými frekvencemi. Detaily obsahují nejvyšší frekvence a při zvětšující se vzdáleností stačí pro popsání nižší frekvence. Po odfiltrování zůstanou struktury s nižší frekvencí, než je zvolené měřítko. Toto lze využívat pro sestavování 3D rekonstrukci scén, právě díky měřítku jsou získány informace k třetí dimenzi. Funkce, která umožňuje odfiltrovat vyšší frekvence je například Gaussova funkce, která se chová jako dolní propust. Vzorec (4.1) je přepisem Gaussovy funkce. V praxi se ovšem využívá spíše derivace a Laplacian Gaussovy funkce (LoG).[16]

$$G\left(X_1, ..., X_N, \sigma\right) = \frac{1}{\left(2\pi\sigma^2\right)^{(N/2)}} e^{\frac{-\left(X_1^2+...+X_N^2\right)}{2\sigma^2}} \tag{4.1}$$

Ve své práci se věnuji dvěma novějším metodám a to SIFT 4.3 a SURF 4.4. Obě metody jsou měřítkově nezávislé, tak, že kromě lokalizace význačných bodů také zjistí jejich měřítko. Jsou založeny na filtrování pomocí derivací Gaussovy funkce.

4.2 Hledání korespondencí

Podle charakterizujících vlastností význačných bodů, lze tyto body opětovně dohledat. Většina algoritmů poskytuje jak seznam klíčových bodů tak i samotný deskriptor. Deskriptory tedy hrají nejdůležitější roli při porovnání více obrázků a nalezení správné korespondence. Kvalita deskriptoru určuje i náchylnost algoritmu ke geometrickým zkreslením.

Ke generování deskriptorů se využívá okolí význačných bodů, z kterých se vytvoří vektor, jehož velikost prvků charakterizují daný bod. Při hledání korespondence se využívají tyto deskriptory obou obrazů, kde se prvky z obou deskriptorů porovnají euklidovskou vzdálenosti. Čím nižší vzdálenost, tím jsou body k sobě podobnější. U porovnávání deskriptorů se vždy nalezne minimální hodnota a to povede k nalezení korespondence, i když mezi obrázky žádná nemusí být. Těmto korespondencím se říká "Tentávní korespondence", ty obsahují páry popisů bodů, jež jsou vzdálenostně podobné, ale neodpovídají jednomu fyzickému prvku. Takovýmto bodům se říká *outliers* (odlehlé hodnoty).[17]

Při hledání korespondencí se používá jeden z těchto postupů:

- Vzájemně nejbližší popisy význačných bodů z prvního obrazu k druhému obrazu
- Stabilní párování popisu. Nalezená korespondence mezi body A a B se z dalšího hledání vyloučí. Korespondence se hledají tak dlouho, pokud zbývají aktivní prvky
- Naleznou se k popisu prvního obrázku dva nejbližší z druhého obrázku. Když je poměr vzdálenosti menší než 0.8, popisy korespondují a je zaručeno, že v okolí není žádný bod, s kterým bychom si ho mohli splést. Tuto metodu lze použít pro diskriminativní popisy, jako je například SIFT

4.3 SIFT

SIFT anglicky znamená Scale-Invariant Feature Transform. Je to algoritmus pro počítačové vidění vztahující se k bodu z různých pohledů na 3D scénu. Používá se pro detekci objektů a popsaní funkcí v obrazu. Algoritmus byl publikován David Lowe v roce 1999.[18]

Pro každý objekt v obraze nalezne klíčové body, které můžou být dále zpracovány pro popis funkce. Tento popis může být použit třeba pro vyhledání určitých objektů. Rozpoznání se může provádět spolehlivě i při změnách měřítka obrazu, šumu a osvětlení. Body se obvykle hledají na vysoce kontrastních oblastech obrazu, například hrany a rohy.[18]

SIFT má velice kvalitní deskriptory, nalezené klíčové body jsou velmi stabilní, je ovšem relativně pomalý. Nalezené body jsou nezávislé na měřítku, rotaci a některým afinním deformacím, šumu a změně osvětlení.

4.3.1 Sestrojení měřítkové nezávislosti a vyhledání lokálních extrémů

Metoda, která vytvoří měřítkově nezávislou reprezentaci celého obrazu, jako sbírku příznakových vektorů, které jsou invariantní k přeloženému snímku, změně měřítka, rotace, osvětlení a lokálního geometrického zkreslení. Poté se vyhledávají všechny lokální extrémy, tyto hlavní míst jsou detekovány jako maxima a minima z rozdílu Gaussových funkcí (DoG). Navržené body podél hran a s nízkým kontrastem jsou vyřazeny, to zajišťuje větší stabilitu klíčových bodů. Cílový výsledek je získán tím, že se zvažují pixely kolem klíčových míst.[18, 19]

4.3.2 Lokalizace klíčových bodů, jejich potvrzení a indexování

Získané body z předchozí fáze se dále testují. Ty, které nejsou dostatečně stabilní, se vyloučí a u zbylých se pomocí interpolace upřesní jejich poloha v prostoru. Indexace se skládá ze získání klíčů a určení odpovídajícího klíče pro následující obraz. Používá se k tomu modifikace KD tree algoritmu s názvem best-bin-first (BBF), kterým lze identifikovat nejbližší sousedy. Zásobník je prohledáván od nejbližší vzdálenosti místa dotazu. Vyžaduje použití haldy na bázi prioritní fronty. Nejbližší sousedé jsou definovány jako klíčové body s minimální euklidovskou vzdálenosti od daného vektoru deskriptoru. Zamítnou se všechny navržené shody, jejichž poměr vzdálenosti je větší než 0.8, což eliminuje 90% falešných a 5%5 správných shod. Hodnota poměru vzdálenosti je vyhodnocována derivací rozdílu Gaussových funkcí v místě zkoumaného klíčového bodu.[18, 19]

4.3.3 Přiřazování orientací, identifikace shluku podle volby Hough transformace

Každému klíčovému bodu je přiřazena orientace na základě orientací gradientů v okolí bodu. To se dělá pro zajištění nezávislosti na rotaci. Houghova transformace odhalí shluky vlastností s jednotným výkladem z každé funkce, pak volí všechny objekty v pozici, které jsou v souladu s funkcí. Když objekty na dané pozici mají stejné shluky vlastností, je pravděpodobnost správnosti výkladu mnohem větší. Položka je v hash tabulce vytvořena podle orientace, umístění a měřítka ze shod hypotéz modelu. V hash tabulce se vyhledává identifikace všech shluků, nejméně se třemi položkami v zásobníku. Zásobník je seřazen sestupně podle velikosti. Každý klíčový bod ze SIFTu určuje umístění ve 2D prostoru, měřítko, orientaci a vztah k obrazu. Podobnostní transformace vychází z těchto čtyř parametrů, jejich aproximací se přibližujeme k 3D scéně, ale nebere v úvahu nerigidní deformace.[18, 19]

4.3.4 Ověření modelu podle lineární metody nejmenších čtverců

Každý identifikovaný shluk je ověřen lineární metodou nejmenších čtverců. Afinní transformace bodu modelu $[xy]^T$ k bodu obrázku $[uv]^T$, může být napsáno následovně (4.2).

$$\begin{bmatrix} u \\ \\ v \end{bmatrix} = \begin{bmatrix} m1 & m2 \\ \\ m3 & m4 \end{bmatrix} \begin{bmatrix} x \\ \\ y \end{bmatrix} + \begin{bmatrix} tx \\ \\ ty \end{bmatrix} \tag{4.2}$$

V rovnici (4.2), kde je vzor překladu $[txty]^T$ a affinní rotace, měřítko a roztažení jsou reprezentovány parametry m1, m2, m3 a m4. Je potřeba minimálně třech shod pro získaní řešení. Minimalizuje to počet čtverců vzdálených od předpokládané lokality na odpovídajícím místu obrazu.[18, 19]

4.3.5 Odhalení odlehlých hodnot

Odlehlé hodnoty mohou být odstraněny mezi kontrolou obrazových funkcí a vzoru, s ohledem na parametry. Vzhledem k použití metody nejmenších čtverců pro každou

shodu, je důležité určit polovinu rozsahu chyb parametru pro Hough transform zásobník. Vyřazené odlehlé body je potřeba znovu vyřešit pomocí zbývajících bodů a ten proces opakovat. Zůstane-li po vyřazení méně jak tři body, shoda bude definitivně vyřazena. Jinak konečné rozhodnutí o přijetí, či odmítnutí hypotéz je založeno na pravděpodobnosti. Pokud je konečná pravděpodobnost správného výkladu větší jak 0,98 je model přijat.[18, 19]

4.3.6 Tvorba deskriptoru

Pro každý významný bod je vypočten deskriptor na základě gradientu orientace, což umožňuje malé posuvy v umístění, z toho důvodu je výsledný algoritmus odolný proti geometrickým změnám a malým deformacím. Významné body jsou pomocí tohoto deskriptoru porovnávány mezi jednotlivými snímky. Všechny kroky SIFTu směřovali k tomu, aby zůstali pouze stabilní a dobře lokalizované body. Deskriptor nese v sobě informace o okolí význačného bodu.

4.4 SURF

SURF je anglická zkratka Speeded Up Robust Features. Je to algoritmus pro vyhledávání významných bodů a následné korespondence mezi dvěma snímky. Byl publikován Herbertem Bayem v roce 2006. SURF se snaží o zjednodušení a urychlení vyhledávání [20]. Používá pro derivaci Gaussovy funkce obdélníkové aproximace a i přesto dosahuje kvalitních výsledků. Spoléhá na integrální obraz a staví na rychlé Hessově matice a rozdělení bází. Deskriptor popisuje okolí zvoleného bodu pomocí Haarovy vlny. Stupeň indexace je založen na znaménku Laplacaina, který zvyšuje rychlost získání odpovídajících shod a robustnost deskriptoru. Dále je samotný deskriptor méně obsáhlý než deskriptor u SIFTu, to snižuje čas výpočtu a porovnávání.[19, 21]

4.4.1 Integrální obraz

Je to struktura vytvořena ze zdrojového obrázku, slouží pro rychlé zjištění součtu hodnot bodů uvnitř libovolného obdélníku zdrojového obrazu. Obdélník má stejné rozměry jak ve zdrojovém tak i v integrálním obrazu, a začíná v horním levém rohu a v libovolném bodu. Po vygenerování integrálního obrazu je možné takřka okamžitě zjistit součet libovolné oblasti původního obrázku pomoci sečtení čtyř číslic. Součet bodů ve vybrané oblasti je pak získán pomocí hodnot bodů v integrálním obrázku, podle vzorce (4.3). Kde "A"je bod v pravém dolním rohu, "B"bod v levém dolním rohu, "C"bod v pravém horním rohu a "D"bod v levém horním rohu obdélníku.[21]

$$\Sigma = A - B - C + D \tag{4.3}$$

4.4.2 Detektor Fast-Hessian

Tento detektor staví na Hessově matici, má dobrý výkon i přesnost. Spoléhá se zde na determinant Hessiánu, jak pro umístění, tak pro rotaci. Hessova matice (4.4) je definována následovně:

$$H(X,\sigma) = \begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ \\ L_{yx}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix} \tag{4.4}$$

V Hessově matici (4.4) je L_{xx},L_{yx},L_{xy} a L_{yy} konvolucí z Gaussova druhé derivace v bodě X a Y. Gausson je optimální pro měřítkovou nezávislost v prostoru, v SURFu se používá celkem radikální aproximace a to pomocí obdélníkových funkcí. Je nutné výsledky normalizovat a to se provádí s Frobeniovou normou. Po vygenerování se význačné body najdou stejně jako u SIFTu a to pomocí lokálních maxim. U vyšších měřítek je potřeba ještě aproximace polohy maxima pomocí Taylorová rozvoje.[21]

4.4.3 Přiřazení orientací

Postup je podobný jako u algoritmu SIFT, na rozdíl od SIFTu se bere v potaz pouze nejdominantnější směr, který se určuje na kruhovém okolí význačného bodu. Pro záznam orientace jsou použity posuvné bubliny orientací, které pokrývají dohromady 360°, jedna bublina pokrývá 60°pro dosažení největší stability. Po analýze všech bodů kolem význačného bodu se vytvoří nový vektor v každé bublině. Vektory se porovnají a vybere se největší z nich, jeho orientace určí pak také orientaci význačného bodu.[21, 22]

4.4.4 Tvorba deskriptoru

Algoritmus SURF přináší nový druh deskriptoru, který je inspirován deskriptorem SIFTu. Zanechává jeho výhody jako rozdělování na subregiony, díky kterým je deskriptor odolný vůči rotacím a geometrickým změnám. U SURFu jsou tyto subregiony jinak zpracovány. Vytvoření deskriptoru začíná rozdělením do čtvercových subregion a v každém se vybere pět pravidelně rozmístěných bodů, které projdou filtrem. Pro každý z těchto bodů se vypočtou odezvy, které jsou přičteny k deskriptoru dané oblasti, který obsahuje čtyři hodnoty. Tyto hodnoty subdeskriptoru se získají pro všech 16 subregionů. Celkový deskriptor význačného bodu je popsán 64 hodnotami. Je vidět, že výsledný deskriptor je poloviční oproti deskriptoru SIFTu. To umožňuje rychlost porovnávání, jeho použitelnost v real-time aplikacích je o něco lepší, i přes trochu menší stabilitu význačných bodů než je u SIFTu.[21]

4.5 RANSAC

RANSAC je zkratka anglických slov RANdom SAmple Consensus. Jedná se o iterační metodu pro odhad parametrů matematického modelu z naměřených údajů, které obsahují *inliears* (správné popisy) a *outliers* (odlehlé hodnoty). RANSAC je nedeterministický

algoritmus, který vytváří rozumné výsledky s určitou pravděpodobnosti správnosti, tato pravděpodobnost se zvyšuje každou iterací. Algoritmus byl publikován Fischler a Bolles v roce 1981.[23]

Získané dvojce bodů z tentativní korespondence můžou a většinou i obsahují nekorespondující body (outliers). Tyto body mají podobné popisy, ale neodpovídají jednomu fyzickému prvku. Cílem RANSACu je zbavit se takovýchto korespondencí a výsledkem je nalezení modelu transformace scény a množinu správných korespondencí (inliers), jenž jsou konzistentní s nalezenými body.[17]

4.5.1 Postup

Množina vstupních hodnot dat, která je určena jako vstup do algoritmu RANSAC musí obsahovat alespoň 4 bezchybných korespondencí, které nejsou na přímce, aby se nalezl model scény dvou snímků. Jestli je tento předpoklad splněný může se pokračovat následujícími kroky:[17, 23]

1. Náhodně se vyberou 4 korespondující páry, hypotetické inliers
2. Se zvolených párů se vypočítá homografie,
3. Zjistí se podpora k modelu, ta se kvalifikuje jako počet bodů konzistentních s modelem,
4. Zda je podpora nalezené homografie nejvyšší, uložíme si její hodnotu homografie a množinu všech inliers párů,
5. Body 1-4 se iterativně opakují, a hledá se taková homografie, která správně transformuje největší počet korespondencí.

4.5.2 Výhody a nevýhody

K výhodam RANSACu patří robustní odhad parametrů modelu s vysokou přesností, ač při stupu měla data velký počet odlehlých hodnot. Nevýhodou je, že neexistuje žádná horní hranice pro čas výpočtu. To se řeší omezením počtu iterací, to může mít za následek neoptimální výsledek. V praxi se používá pro zastavení iterací odhad pravděpodobnosti nalezení správného vzorku.[23]

4.6 OpenCV

OpenCV je otevřená multiplatformní knihovna pro práci a manipulaci s obrazem. Je zaměřená na zpracování obrazu v reálném čase. OpenCV je napsaná v C/C++ a lze ji využívat pomocí jazyků C, C++, Python, Java a další. Oficiálně byl projekt OpenCV zahájen v roce 1999.[14]

OpenCV je silně zaměřená na real-time aplikace a podporuje více jádrové procesory. Ve volně dostupné knihovně jsou k dispozici více jak 500 algoritmů pro počítačové vidění. V bakalářské práci budu využívat funkce a algoritmy implementované v této knihovně.

4.6.1 Instalace OpenCV na Ubuntu

Instalace OpenCV na Ubuntu není nijak složitá. Stačí udělat tyhle následující kroky.

1. Stáhneme se OpenCV 2.4.6.1, třeba z tohoto odkazu `http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.6.1/opencv-2.4.6.1.tar.gz/download`. A poté ho rozbalíme do složky, kde s ním budeme chtít pracovat. V rozbalených datech nalezneme i všechny další potřebné kroky k dokončení instalace OpenCV.

2. OpenCV potřebuje podobné knihovny jako SDK, nicméně budeme muset ještě doinstalovat aplikaci *CMAKE* a knihovny *python-dev, python-numpy, libopencv-dev a libgtk2.0-dev*[15]

3. Vytvoříme složku, kde následně provedeme příkaz *cmake <USE_TRIP_PATH>*. Parametr programu je cesta ke souborům OpenCV (složka, kde jsem rozbalili OpenCV). Do aktuální složky se vytvoří potřebné soubory ke kompilaci.

4. Ve složce je i soubor *Makefile*. Proto provedeme příkaz *make*. Kompilace bude chvíli trvat tak si můžeme zajít na kafe.

5. Po úspěšném dokončení provedeme ještě příkaz *sudo make install*. Tento příkaz nám přidá knihovny od OpenCV do systémových (jejich odkazy) a usnadní nám to pozdější práci. [15]

5 Implementace

V předešlých kapitolách jsem popisoval teorii k detekci význačných bodů a jejich spárování 4 kapitola, ale také samotný letoun 2 kapitola a prostředí 3 kapitola, ve kterém jsem pracoval. Pro samotnou implementaci jsem si zvolil programovací jazyky C a Python. Cílová aplikace je určena pro systém GNU/Linux, ve kterém jsem dělal, abych byl přesnější tak to bylo Ubuntu 12.04 LTS.

5.1 Struktura projektu

Má bakalářská práce je rozdělena do dvou samostatných programů. První program je zaměřený na připojení a ovládání Ar.Dronu2 a získaní Navdat s videem. Tento program je implementován v jazyku C, s využitím knihoven ARDrone_SDK. Úvod k SDK byl popsán v 3 kapitole. Druhý program se připojí k video streamu a zpracovává jednotlivé snímky, naměřené údaje pak uloží do patřičných souborů. Tento program jsem implementoval v programovacím jazyce Python pomocí OpenCV. Teoretická část zpracování obrazu je probrána v 4 kapitole.

Oba programy jsou na sobě nezávislé a můžou pracovat samostatně. K programu na zpracování obrazu, lze připojit jiné video, pokud ovšem chceme zpracovávat video z kvadkoptéry je nutné se k ní připojit. A to pomocí systémových nástrojů k připojení bezdrátové sítě a použitím třeba prvního programu.

5.2 Program na ovládání kvadkoptéry

Tento program se zabývá připojením, komunikací a ovládání Ar.Dronu2. V této kapitole je popsáno rozdělení zdrojových kódu. Pro vytvoření této části aplikace bylo použito programovacího jazyka C s knihovnami ARDrone_SDK. Program využívá práci s více vlákny.

Vlákna nám zajišťují střídání mezi více úlohami, které mají společné proměnné a data. Makra pro práci s vlákny jsou definované v souboru *vp_api_thread_helper.h*, který nám poskytuje ARDrone_SDK. Nejdůležitější makra jsou *START_THREAD*, *JOIN_THREAD*, *DEFINE_THREAD_ROUTINE* a *BEGIN_THREAD_TABLE*. Tyto makra definují, spouští, čekají na ukončení vlákna, poslední z nich přidává záznam do manageru vláken a každé z vláken musí mít vyplněnou svojí strukturu, kde se nastavuje název vlákna a odkazy na funkce pro otevření, změnu a ukončení. V 5 výpisu vidíte naplnění struktury pro vstupní zařízení gamepad.

5.2.1 Hlavní vlákno

Nastaví a inicializuje všechny potřebné proměnné, parametry, zařízení a připojení se k Ar.Dronu. Musí obsahovat makro *BEGIN_THREAD_TABLE*, jeho příklad můžete vidět ve 3 výpisu. Toto vlákno také musí obsahovat funkce *ardrone_tool_init_custom(void)* a *ardrone_tool_shutdown_custom(void)*. První nastavuje konfigurace Ar.Drona a předvolby přijímaných dat a Navdat, přidaní a inicializace vstupních zařízení a také samotné spuštění

jednotlivých vláken. Druhá funkce se volá při ukončování aplikace a čeká na ukončení všech vláken, odpojuje vstupní zařízení a odpojení od Ar.Drona.

```
BEGIN_THREAD_TABLE
    THREAD_TABLE_ENTRY(video_stage, 20)
    THREAD_TABLE_ENTRY(video_recorder, 20)
    THREAD_TABLE_ENTRY(navdata_update, 20)
    THREAD_TABLE_ENTRY(ardrone_control, 20)
    THREAD_TABLE_ENTRY(gtk, 20)
END_THREAD_TABLE
```

Výpis 3: Macro: Tabulka vláken

5.2.2 Vlákno pro Navdata

Tohle vlákno obsahuje pouze tři funkce. Inicializační funkci, kde se může nastavit konfigurace Ar.Drona, aby posílal Navdata takové, jaké si budeme přát. Ukončovací funkci a funkci pro průběh příjmu Navdat, tato funkce může obsahovat uložení Navdat do vlastní struktury, či jejich vypsání (zobrazení). Odkazy na tyto tři funkce musí obsahovat makro *BEGIN_NAVDATA_HANDLER_TABLE*. Příklad tohoto makra vidíte ve 4 výpisu. Toto makro musí být umístěno v hlavním vlákně anebo ve vlákně pro Navdata, jiné umístění způsobí kolize a špatné zpracování.

```
BEGIN_NAVDATA_HANDLER_TABLE
    NAVDATA_HANDLER_TABLE_ENTRY(navdata_init, navdata_process, navdata_release, NULL)
END_NAVDATA_HANDLER_TABLE
```

Výpis 4: Macro: Tabulka navdat

5.2.3 Vlákno pro ovládání

V hlavním vlákně musí být inicializace vstupního zařízení pomocí ID, které bylo ve výpisu z příkazu *lsusb*. Toto vlákno jsem implementoval pomocí příkladu, které byli v archívu společně s ARDrone_SDK. Funkci pro přidání vstupního zařízení a kontrolu ID pomocí LL1 gramatiky ID jsem ponechal původní. Přidal jsem pouze funkce pro otevření, ukončení a ovládání. Funkce ovládání se provádí každých 20 ms, vždy projde buffer zmáčknutých hodnot, které se zíkají od zařízení. Ty zhodnotí a podle získaných informací a ARDrone_SDK pošle AT příkazy pohybu na drona. AT příkazy byli probrány v 2.4.1 kapitole.

```
input_device_t  gamepad = {
    "PS3Gamepad",
    open_gamepad,
    update_gamepad,
    close_gamepad };
```

Výpis 5: Příklad naplnění struktury pro vstupní zařízení

5.2.4 Vlákno pro video

Pro příjem videa nestačí nastavit pouze strukturu s odkazy na funkce zpracování, ale musí se nastavit i vstupní a výstupní obrázek. To se provádí ve funkci *ardrone_tool_init_custom* v hlavním vlákně. Musí se nastavit z jaké kamery drona se budou přijímat snímky, jaké mají kódování, rozlišení a snímkovaní. Musí se také vytvořit zásobníky pro tyto obrázky. Všechny tyto proměnné se pak předají samotnému vláknu pro video, ukázka předaní v 6 výpisu.

V tomto vlákně také ukládám samotné video do souboru *"ArDrone-video.avi"*, do složky *Data* v adresáři aplikace.

```
START_THREAD(video_stage, (specific_parameters_t *) params);
    video_stage_init () ;
```

Výpis 6: Spuštění vlákna videa

5.3 Program na zpracování obrazu

Tento program se zabývá zpracováním snímku z Ar.Drona. Zdrojový kód je napsán v programovacím jazyce Python, pomocí knihoven OpenCV. Teoretická část je popsána v 4 kapitole.

Vyzkoušel jsem vyhledávat význačné body obrazu z fotografií tak z videa, za pomocí metod SIFT a SURF. Obě metody poskytly vždy stovky až tisíce bodů. Nakonec jsem se rozhodl pro algoritmus SIFT, který poskytuje stabilnější deskriptor. Tyto deskriptory fungovali spolehlivě při hledání korespondence mezi jednotlivými snímky. Občas se vyskytli chybné korespondence, převážně v oblastech, kde byli podobné objekty.

Tento program má dvě vlákna. Jedno se připojí na video stream Ar.Drona a druhé jednotlivé snímky zpracovává a ukládá naměřené údaje.

5.3.1 Připojení k video streamu

Nejdříve se připojíme pomocí OpenCV v Pythnu k video streamu prvním příkazem z 7 výpisu. Poté se nastaví vlákno s odkazem na metodu, která se bude provádět, a pak se toto vlákno spustí.

```
self.video = cv2.VideoCapture(tcp://192.168.1.1:5555)
self.thread = threading.Thread( target=self.my_run )
self.thread.start ()
```

Výpis 7: Připojení na video stream Ar.Drona2

Metoda *my_run* z 8 výpisu získá následující snímek z videa a nastaví ho do atributu třídy *self.image*. Po projití cyklu se vždy přepne na druhé vlákno, a to díky příkazu *time.sleep(0.001)*. Vlákna v Pythnu fungují trochu jinak než v C-éčku a je zapotřebí jednotlivé vlákna uspávat, jinak se nepřepne na jiné vlákno a bude se provádět stále jen to jedno.

```
def my_run(self):
    while self.running:
        ret, image = self.video.read()
        if ret:
            self.image = image[:]
        else:
            self.image = None
            time.sleep(0.001)
    print "Video vypnuto"
```

Výpis 8: Získávání snímku ze streamu

5.3.2 Zpracování obrazu

OpenCV nabízí několik algoritmů na detekci význačných bodů, já si vybral algoritmus SIFT. Teorie k této problematice byla popsána v 4.3 kapitole. Pro hledání korespondencí používáme metodu RANSAC.

```
detector = cv2.SIFT()
norm = cv2.NORM_L2
matcher = cv2.BFMatcher(norm)
```

Výpis 9: Inicializace detektoru pro zpracování obrazu

```
kp1, desc1 = detector.detectAndCompute(image1, None)
kp2, desc2 = detector.detectAndCompute(image2, None)
```

Výpis 10: Detekce význačných bodů

Když nastavíme detektor pro získání význačných bodů a metodu na nalezení korespondence mezi dvěma snímky, například jak je tomu ve 9. Můžeme hned používat metody těchto objektů. SIFT nabízí několik metod pro nalezení význačných bodů a deskriptoru, já používám metodu, která dá jak pole význačných bodů tak samotné deskriptory k těmto bodům (10 výpis). Toto pole a deskriptor uložím vždy do samostatného souboru s názvem *siftN.txt*, tento soubor je umístěn do složky *./Data/sift/*. Jednotlivé snímky ukládám do složky *./Data/img/* pod názvy *imgN.png*. Písmeno N v názvu znamená celé kladné číslo.

```
raw_matches = matcher.knnMatch(desc1, desc2, k = 2)
point1, point2, kp_pairs = my_filter_matches(raw_matches)
if len(point1) >= 4:
    homograph, status = cv2.findHomography(point1, point2, cv2.RANSAC, 10.0)
```

Výpis 11: Korespondence mezi snímky

Teorie k hledání korespondencí je v kapitolách 4.2 a 4.5. Nejdříve získáme všechny shody, kde jsou i nesprávné shody, které jsou si pouze podobné. Proto projdeme celý seznam filtrem (12), který projde jednotlivé dvojce a podle poměru buď zanechá shodu v seznamu anebo ji vyřadí. Pokud nám zůstanou čtyři a více páru můžeme přejít k samotnému RANSACu, který nám pomocí zbylých klíčových bodů z obou obrázků vyhledá

homograp a statusy k dvojicím bodů. Tento postup je znázorněn ve 11 výpisu. Do souboru *ransacN.txt* uložíme seznam dvojic klíčových bodů, homograp a statusy, soubor je umístěn ve složce *./Data/ransac/*.

```
def my_filter_matches(kp1, kp2, raw_matches, ratio = 0.75):
    mkp1, mkp2 = [], []
    for m in raw_matches:
        if len(m) == 2 and m[0].distance < m[1].distance * ratio:
            m = m[0]
            mkp1.append( self.kp1[m.queryIdx] )
            mkp2.append( self.kp2[m.trainIdx] )
    p1 = np.float32([kp.pt for kp in mkp1])
    p2 = np.float32([kp.pt for kp in mkp2])
    kp_pairs = zip(mkp1, mkp2)
    return p1, p2, kp_pairs
```

Výpis 12: Filtr pro korespondenci

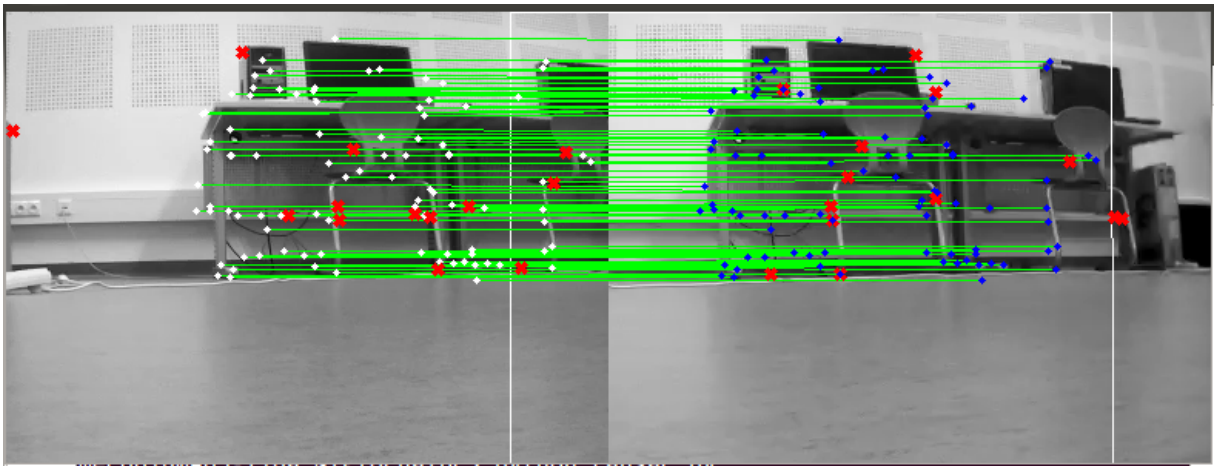
5.4 Testování

Implementované programy jsem testoval průběžně ve škole i doma. Testoval jsem hlavně program na připojení k Ar.Dronu2, jeho ovládání a uložení videa. U zpracování obrazu se zdálo být vše v pořádku, už od začátku implementace. SIFT o snahu, co nestabilnější a nejpřesnější lokalizaci význačných bodů , je pro větší rozlišení celkem pomalý. Testoval jsem to na stolním počítači s jedno jádrovým procesorem AMD Sempron(tm) Processor 3200, s frekvencí 1800 MHz a operační pamětí 939,0 MiB, zpracování jednoho obrázku trvalo od 0.1 až do 0.45 sekund.

Zvýšením odstupu první a druhé minimální vzdálenosti deskriptoru, lze omezit chyby korespondencí. V mém případě to lze udělat ve funkci *my_filter_matches* parametrem *ratio*, nastavením na menší hodnotu.

Na obrázku 5.1 vidíte dva snímky, první vlevo a druhý vpravo. Použitím programu, pro zpracování obrazu dostáváme výsledek, který je znázorněný v obrázku 5.2. Červené křížky značí vyloučené korespondence pomocí RANSACu, zelená úsečka odpovídá správné korespondence mezi jednotlivými páry význačných bodů, ty jsou znázorněny bíle a modře. Bílý rámeček označuje umístění prvního snímku, jak by byl umístěný vůči druhému snímku.

Obrázek 5.1: Originál obrázku, před zpracováním



Obrázek 5.2: Zobrazeni výsledku po RANSACu

Závěr

Počítačové vidění je vyvíjející se vědecká disciplína. A cílem mé práce bylo seznámení se s algoritmy pro vyhledání význačných bodů v obraze kvadkoptéry a jejich následné spárování. Bylo nutné zprovoznit kvadkoptéru Ar.Drone2 od firmy Parrot. Komunikace a ovládání Ar.Dronu je popsáno v kapitolách 2.3 a 2.4. O samotné implementaci pak píší v 5 kapitole. Tato kapitola popisuje jednotlivé části aplikace, která je rozdělena do dvou samostatných programů.

U nových metod pro hledání význačných bodů se klade důraz na nezávislosti na měřítku.Nejdříve jsem nastudoval teorii kolem této problematiky. Mezi dvě nové metody patří SIFT a SURF. SIFT nabízí velmi kvalitní výsledky a nejstabilnější významné body, ale vyhledávání může být pomalé. SURF je rychlejší a nabízí slušné výsledky, ale stabilita významných bodů není tak velká jako u SIFTu.

Metody jsem testoval na snímcích získané z Ar.Drone2 kvadkoptéry. Testy ukázaly, že metodou SIFT z jednoho a toho samého snímku, získáme ty samé význačné body. Rychlost na počítači, kterém jsem to testoval bylo od 0.1 až do 0.45 sekund. Vyhledávání význačných bodů je základem většiny aplikací zabývající se počítačovým viděním.

Přínosem mé práce je uložení všech užitečných dat, které jsou připraveny k dalšímu zpracování k 3D rekonstrukci. Tento program může být rozšiřován v navazující práci dalšího studia. Aplikace umožňuje ovládat kvadkoptéru základními směry a rotacemi, není navržena pro akrobatické kousky. Práce se skládá ze dvou samostatných programů, lze je spustit nezávisle na sobě. To nám umožňuje ovládat Ar.Drona aniž bychom zpracovávali obraz a naopak, můžeme zpracovávat video, snímky z předešlé činnosti anebo úplně z jiných zdrojů.

Miroslav Meca

Literatura

[1] UAV gizmag.com *The UAV* [Online]. ©2014 [cit. 2014-04-15]. Dostupné z: http://www.gizmag.com/tag/uav/

[2] The UAV *The UAV* [Online]. ©2012 [cit. 2014-04-15]. Dostupné z: http://www.theuav.com/

[3] AR.Drone open API platform *ArDrone API, ARDrone_SDK_2_0_1_Developer_Guide.pdf* [Online]. ©2009 [cit. 2014-04-10]. Dostupné z: https://projects.ardrone.org/attachments/download/514/ARDrone_SDK_2_0_1.tar.gz

[4] Ar.Drone2 PARROT. *AR.Drone 2.0 Parrot quadcopter* [Online]. ©2013 [cit. 2014-04-15]. Dostupné z: http://ardrone2.parrot.com/

[5] Technical specification Ar.Drone2 PARROT. *AR.Drone 2.0 Parrot quadcopter* [Online]. ©2013 [cit. 2014-04-15]. Dostupné z: http://ardrone2.parrot.com/ardrone-2/specifications/

[6] Technická specifikace. PARROT. *Parrot Ar.Drone2 Cz* [Online]. ©2012 [cit. 2014-04-16]. Dostupné z: http://ardrone2.parrot-ardrone.cz/specifications/

[7] LiPo akumulátory. BATTEX. *Li-ion akumulátory v praxi* [Online]. ©2014 [cit. 2014-04-16]. Dostupné z: http://www.battex.info/hermeticke-akumulatory/li-akumulatory/pouzivani-li-ion-akumulatoru-v-praxi

[8] A MEMS Clearinghouse. MNX. *Co je MEMS technologie* [Online]. [cit. 2014-4-16]. Dostupné z: http://www.memsnet.org/mems/what_is.html

[9] Gyroskop. WIKIPEDIA. *Gyroscope* [Online]. ©2014 [cit. 2014-04-17]. Dostupné z: http://en.wikipedia.org/wiki/Gyroscope

[10] Akcelerometr. WIKIPEDIA. *Accelerometer* [Online]. ©2014 [cit. 2014-04-17]. Dostupné z: http://en.wikipedia.org/wiki/Accelerometer

[11] Sdk. GAUTH.FR. *Introduction to the ArDrone SDK* [Online]. ©2010 [cit. 2014-04-18]. Dostupné z: http://gauth.fr/2011/09/introduction-to-the-ar-drone-sdk/

[12] Sixaxis Help Ubuntu. *Connecting joystick SONY PS3, Sixaxis or DualShock 3* [Online]. ©2013 [cit. 2014-04-18]. Dostupné z: https://help.ubuntu.com/community/Sixaxis

[13] Harris affine region detector. WIKIPEDIA. *Harris affine region detector* [Online]. ©2014 [cit. 2014-04-29]. Dostupné z: http://en.wikipedia.org/wiki/Harris_affine_region_detector

[14] OpenCV. *OpenCV* [Online]. ©2014 [cit. 2014-04-19]. Dostupné z: http://opencv.org/

[15] OpenCV on Ubuntu. LAZ'S VISION. *Install OpenCV 2.4 on Ubuntu 12.04* [Online]. ©2013 [cit. 2014-04-19]. Dostupné z: http://karytech.blogspot.cz/2012/05/opencv-24-on-ubuntu-1204.html

[16] D. G. Lowe. Object recognition from local scale-invariant features. *In Proceedings of International Conference on Computer Vision*. Strana 1150–1157, 1999.

[17] Hledání korespondencí. FELK CVUT. *Hledání korespondencí* [Online]. ©2013 [cit. 2014-04-30]. Dostupné z: https://cw.felk.cvut.cz/wiki/courses/a4m33mpv/cviceni/2_hledani_korespondenci/start

[18] SIFT. SCHOLARPEDIA. *SIFT* [Online]. ©2012 [cit. 2014-04-19]. Dostupné z: http://www.scholarpedia.org/article/SIFT

[19] SIFT. WIKIPEDIA. *SIFT* [Online]. ©2014 [cit. 2014-04-19]. Dostupné z: http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

[20] SURF. WIKIPEDIA. *SURF* [Online]. ©2014 [cit. 2014-05-01]. Dostupné z: http://en.wikipedia.org/wiki/SURF

[21] SURF. Herbert Bay, Tinne Tuytelaars and Luc Van Gool *Speeded Up Robust Features* [Online]. [cit. 2014-05-01]. Dostupné z: http://www.vision.ee.ethz.ch/~surf/eccv06.pdf

[22] SURF OpenCV. *Introduction to SURF* [Online]. ©2014 [cit. 2014-05-01]. Dostupné z: http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html

[23] RANSAC. WIKIPEDIA. *RANSAC* [Online]. ©2014 [cit. 2014-05-02]. Dostupné z: http://en.wikipedia.org/wiki/RANSAC

Přílohy

CD-R - obsahuje tento dokument v pdf formátu a archív se zdrojovými kódy